Proceedings of The Mediterranean Conference on Control and Automation, June 16-19, 2015, Malaga, Spain

# Defensive State Feedback Control of Asynchronous Sequential Machines

Jacob Hammer

***Abstract*— The design of state feedback controllers that protect asynchronous sequential machines from pre-programmed adversarial agents is considered. Necessary and sufficient conditions for the existence of such controllers are derived. These conditions are stated in terms of certain matrices of zeros and ones derived from the given description of the protected machine. Controller design algorithms are outlined.**

***Index Terms*— malware, pre-programmed agents, automatic feedback control, asynchronous sequential machines**

## I. Introduction

Over the last few decades, computing systems and networks have experienced a growing threat from pre-programmed adversarial agents that attempt to corrupt the systems and subvert their operation. At the same time, demands for speedy performance have substantially increased the prevalence of asynchronous sequential machines in computing and networking systems (e.g., [1], [2], [3]). In this paper, we develop a control theoretic approach to overcome threats posed by pre-programmed adversarial agents on asynchronous sequential machines. We derive necessary and sufficient conditions for the existence of automatic feedback controllers that counteract such agents; controller construction is also outlined. Particular attention is placed on assuring deterministic behavior of the controlled machines.

Specifically, we consider the configuration of Figure 1, where two controllers – an adversarial controller $C_A$ and a defensive controller $C_D$ – act on the asynchronous machine $\Sigma$. Both controllers are autonomous state feedback controllers, and each accesses $\Sigma$ through its own input channel. Here, $C_A$ and $C_D$ are pre-programmed asynchronous machines; the composite machine of Figure 1 is denoted by $\Sigma(C_A, C_D)$.

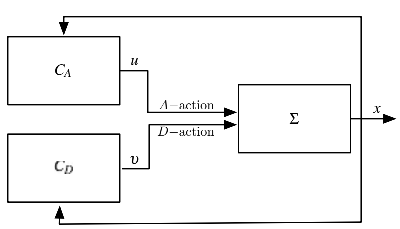


Fig. 1. Adversarial/defensive control

The controllers $C_A$ and $C_D$ act alternately, each applying in its turn a string of characters to $\Sigma$. The objective of $C_A$ is to drive $\Sigma$ into a *target set* of states $T_A$, while the objective of $C_D$ is to drive $\Sigma$ into a *target set* of states $T_D$, where $T_A$ and $T_D$ are specified disjoint sets. A controller *prevails* when $\Sigma$ reaches a member of its target set. Sections IV and V present necessary and sufficient conditions for each one of the controllers to prevail. The conditions are expressed in terms of quantities derived from the given description of the controlled machine $\Sigma$.

Later (section V) we characterize states of $\Sigma$ from which one of the controllers cannot be prevented from prevailing. In all cases, we outline controller designs. Like the machine $\Sigma$, the controllers are asynchronous machines, and thus special precautions are required to ascertain deterministic behavior, as discussed next.

### A. Trigger machines

An *asynchronous trigger machine* receives as input short pulses, or *triggers* (ideally, of zero duration, e.g., [4]). As asynchronous trigger machines are in wide use – in fact, most asynchronous computing machines are trigger machines – we concentrate in this note on such machines.

The machine $\Sigma$ of Figure 1 is an asynchronous trigger machine $\Sigma = (A, D, X, f, x_0)$ with two inputs and state output; here, $A$ and $D$ are input alphabets; $X$ is the state set; the partial function $f : X \times (A \times D) \to X$ is the *recursion function*; and $x_0$ is the initial state. In response to a string of trigger input pairs $(u_0, \upsilon_0)(u_1, \upsilon_1)(u_2, \upsilon_2) \cdots \in (A \times D)^+$, the machine $\Sigma$ generates a string of states $x_0 x_1 x_2 \cdots \in X^+$ given by

$$x_{k+1} = f(x_k, (u_k, \upsilon_k)), k = 0, 1, 2, ... \quad (1)$$

A triplet $(x, (u, \upsilon)) \in X \times (A \times D)$ is a *valid combination* if $f$ is defined at it. A stable combination is a valid combination $(x, (u, \upsilon))$ for which $x = f(x, (u, \upsilon))$, namely, $\Sigma$ rests at $x$; in such case, $x$ is a *stable state*. Otherwise, when $x \neq f(x, (u, \upsilon))$, then $x$ is a *transient state*. A transient state is traversed very quickly (ideally, in zero time).

*Notation 1.1: (i)* The symbol '$\neg$' indicates the absence of a trigger. Thus, $(u, \neg) \in A \times D$ means that $u$ is triggered in $A$ with no trigger in $D$, while $(\neg, \neg)$ means no trigger at all. *(ii)* The state set of $\Sigma$ is always $X = \{x^1, x^2, ..., x^n\}$. A subset $S = \{x^{i_1}, x^{i_2}, ..., x^{i_q}\} \subseteq X$ is identified with the set of integers $S = \{i_1, i_2, ..., i_q\}$. □

### B. Fundamental mode operation

Fundamental mode operation (e.g., [4]) is an operating policy that assures deterministic outcomes in an asynchronous environment. In fundamental mode operation, no more than one signal is triggered at a time; this is because, in an asynchronous environment, 'simultaneous' triggers almost always appear sequentially in unpredictable order, leading potentially to an unpredictable outcomes. Similarly,

Jacob Hammer is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA hammer@mst.ufl.edu

no input triggers are allowed while a machine is in transition, since the state at which such triggers reach the machine is unpredictable. Starting at a stable state $x$, the machine $\Sigma$ is operated by a trigger $(u,\upsilon) \in (A \times \neg)$ or $(u,\upsilon) \in (\neg \times D)$; this takes $\Sigma$ into a chain of transitions

$$x_1 = f(x,(u,\upsilon)), x_2 = f(x_1,(\neg,\neg)), x_3 = f(x_2,(\neg,\neg)), \cdots$$

As $\Sigma$ has no infinite cycles, this chain terminates, i.e.,

$$x_i = f(x_i,(\neg,\neg)) \tag{2}$$

for an integer $i \geq 1$. Then, $x_i$ is the *next stable state* of the triplet $(x,(u,\upsilon))$. The *stable recursion function* $s : X \times (A \times D) \to X$ of $\Sigma$ assigns to every valid combination $(x,(u,\upsilon))$ its next stable state $x'$, i.e., $s(x,(u,\upsilon)) := x'$.

For the machine $\Sigma$ of Figure 1, the following are required for fundamental mode operation.

*Rule 1.2:* **Fundamental mode operation.** Starting at a stable state $x$, the machine $\Sigma$ is activated by a string of input triggers $(u_1,\upsilon_1)(u_2,\upsilon_2)... \in (A \times D)^+$. Then, *(i)* for all $i = 1,2,...,$ either $u_i = \neg$ or $\upsilon_i = \neg$; and *(ii)* no input triggers appear while $\Sigma$ is in transition. □

In fundamental mode operation, an input string $(u,\upsilon) = (u_1,\upsilon_1)(u_2,\upsilon_2)\cdots(u_j,\upsilon_j)$ is applied one character at a time starting at a stable state $x$; after each input trigger, we wait for $\Sigma$ to reach a stable state, before applying the next input trigger. At the end of the input string, $\Sigma$ reaches the stable state $x'$. Such a transition from a stable state $x$ to a stable state $x'$ forms a *stable transition*. Users are affected only by stable transitions, since transients are very quick. Therefore, controller design aims at achieving suitable stable transitions of the closed loop machine.

A state $x'$ is *stably reachable* from a state $x$ if there is a stable transition from $x$ to $x'$. Fundamental mode operation of Figure 1 implies the following.

*Rule 1.3:* For the machines $\Sigma, C_A$, and $C_D$ of Figure 1,
*(i)* Only one of the machines may trigger at a time;
*(ii)* A machine must be in a stable state when triggered;
*(iii)* Only one machine can be in transition at a time. □

The following terminology is used below.

*Definition 1.4:* Let $x$ and $x'$ be two states of a machine $\Sigma$ that has the two inputs $A$ and $D$.
*(i)* $A-$*action* is a string of triggers applied in fundamental mode operation to input $A$, with input $D$ inactive.
*(ii)* $D-$*action* is a string of triggers applied in fundamental mode operation to input $D$, with input $A$ inactive.
*(iii)* $x'$ is *stably reachable* from $x$ *by* $A-$*action* if $A-$action can induce a stable transition from $x$ to $x'$.
*(iv)* $x'$ is *stably reachable* from $x$ *by* $D-$*action* if $D-$action can induce a stable transition from $x$ to $x'$. □

### C. Controller turns

The controllers $C_A$ and $C_D$ of Figure 1 operate in turns: in each turn, one controller acts, while the other controller rests in a stable state. The objective of each controller is to drive $\Sigma$ to a target state, or – if this is impossible in that turn – to a 'favorable state'. 'Favorable states' are defined by assigning a weight to each state of $\Sigma$. Then, in each turn, $C_A$ takes $\Sigma$ to a lowest weight stably reachable state, while $C_D$ takes $\Sigma$ to a highest weight stably reachable state.

Such a framework is relevant in many applications. For instance, consider a computing system that controls power plants. In somewhat simplistic terms, an adversarial agent will attempt to bring the system to a state of lowest power production, while a defensive controller will attempt to take the system to a state of highest (or normal) power production.

### D. General background

This paper is within the framework for the control of asynchronous machines of [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], and [15].

There is an extensive literature on the control of finite state sequential machines, including [16], [17], [18], [19], [20], [21], [22], the references cited in these publications, and many others. The studies mentioned in this paragraph do not address issues specific to the operation of asynchronous sequential machines, such as the distinction between stable and transient states or the requirement of fundamental mode operation.

## II. Basics

### A. State weights

As indicated earlier, each controller of Figure 1 attempts in its turn to take $\Sigma$ to a state most favorable to that controller's objectives. The following notion formalizes the term 'most favorable' ($Z$ denotes the integers).

*Definition 2.1:* Let $\Sigma$ be an asynchronous trigger machine with the state set $X$ and the target sets $T_A$ and $T_D$. A function $\omega : X \to Z$ is a *weight function* if $T_A$ is the set of states at which $\omega$ is minimal, while $T_D$ is the set of states at which $\omega$ is maximal. The *weight* of a state $x \in X$ is $\omega(x)$. □

The operating policy of Figure 1 is then as follows.

*Rule 2.2:* In its turn, $C_A$ takes $\Sigma$ to a lowest weight state stably reachable by $A-$action, and stops; $C_D$ takes $\Sigma$ to a highest weight state stably reachable by $D-$action, and stops. □

It is possible for a machine $\Sigma$ to have several stably reachable states of the same extremal weight. In such case, for best efficiency, controller turns terminate at the first encountered stably reachable extremal weight state.

*Rule 2.3:* **Operating policy.** In Figure 1, let $T_A$ and $T_D$ be the adversarial and defensive target sets, respectively, and let $\omega : X \to Z$ be the weight function of $\Sigma$. Then, $C_A$ and $C_D$ operate in alternate turns according to:
*(i)* Start: At the initial state, $C_A$ acts first.
*(ii)* Turns: In its turn, $C_A$ uses $A-$action to drive $\Sigma$, stopping at the first lowest weight stably reachable state it meets; $C_D$ uses $D-$action to drive $\Sigma$, stopping at the first highest weight stably reachable state it meets.
*(iii)* Progression: *(a)* A controller activates when the other controller has reached the end of its turn. *(b)* A controller forfeits its turn if all states of $\Sigma$ it can stably reach are target states of the other controller.
*(iv)* Target states: both controllers remain in a stable state, once $\Sigma$ has reached a target state. □

In particular, the rule implies that, when a controller initiates a turn at a most favorable state it can stably reach, it will end its turn with no action, leaving $\Sigma$ at that state.

*Example 2.4:* Consider the stable state machine $\Sigma = (A,D,X,s,x_0)$, with $A = \{a^1,a^2\}$, $D = \{d^1,d^2\}$, $X = \{x^1,x^2,x^3,x^4,x^5\}$, $x_0 = x^1$, and stable recursion function $s$ of Table I. In the first turn, $C_A$ takes $\Sigma$ to $x^4$. In the next turn, $x^1$ and $x^2$ are the highest weight stably reachable states by $D-$action. The designer of $C_D$ decides to which of these states to take $\Sigma$. Here, selecting $x^2$ would permit $C_A$ to prevail in the upcoming turn – a poor choice. Selecting $x^1$ results in an infinite cycle of the closed loop machine $\Sigma(C_A,C_D)$ with no controller prevailing. □

| $x^i$ | $(a^1,\neg)$ | $(a^2,\neg)$ | $(\neg,d^1)$ | $(\neg,d^2)$ | $\omega(x^i)$ | Target set |
|---|---|---|---|---|---|---|
| $x^1$ | $x^1$ | $x^4$ | $x^1$ | $x^2$ | 4 | |
| $x^2$ | $x^1$ | $x^3$ | $x^2$ | $x^1$ | 4 | |
| $x^3$ | $x^1$ | $x^1$ | $x^4$ | $x^5$ | 1 | $\in T_A$ |
| $x^4$ | $x^4$ | $x^1$ | $x^1$ | $x^2$ | 3 | |
| $x^5$ | $x^1$ | $x^4$ | $x^1$ | $x^2$ | 5 | $\in T_D$ |

TABLE I

STABLE TRANSITIONS OF $\Sigma$

Rule 2.3 implies the following characterization of the possible outcomes of the control process of Figure 1.

*Proposition 2.5:* For Figure 1, *(i)* and *(ii)* are equivalent.
*(i)* The control process terminates with $\Sigma$ in a stable state $x$.
*(ii)* $x$ is a target state; or all states stably reachable from $x$ by both $A-$action and $D-$action have the same weight as $x$.

### B. The local sink

Rule 2.3*(ii)* leads to the following notion.

*Definition 2.6:* Let $\Sigma$ be an asynchronous trigger machine with the state set $X$ and the weight function $\omega$. For a state $x \in X$, let $S(x,A) \subseteq X$ (respectively, $S(x,D) \subseteq X$) be the set of all states that are stably reachable from $x$ by $A-$action (respectively, by $D-$action). The *local* $A-$*sink* $S_A(x)$ (respectively, $D-$*sink* $S_D(x)$) is the set of lowest (respectively, highest) weight states stably reachable from $x$ by $A-$action (respectively, $D-$action), i.e.,

$$S_A(x) := \{x' \in S(x,A) : \omega(x') \leq \omega(x'') \text{ for all } x'' \in S(x,A)\},$$
$$S_D(x) := \{x' \in S(x,D) : \omega(x') \geq \omega(x'') \text{ for all } x'' \in S(x,D)\}.$$

*Example 2.7:* Example 2.4 yields $S_D(x^4) = \{x^1,x^2\}$. □

In view of Rule 2.3*(ii)*, the controllers $C_A$ and $C_D$ must determine whether $\Sigma$ has reached a member of the local sink. This is achieved by state feedback, as follows.

*Lemma 2.8:* Refer to Figure 1. By using state feedback, a controller can determine whether (and at what state) the other controller's turn has ended.

*Proof: (sketch)*. The controller $C_A$ ends its turn when $\Sigma$ reaches a stable state at the first member of $S_A(x)$ it encounters. As $S_A(x)$ is known from the given description of $\Sigma$, state feedback allows $C_D$ to detect when $\Sigma$ stably reaches a member of $S_A(x)$, indicating the end of the $C_A$ turn. The case of $C_D$ action is analogous. ■

Lemma 2.8 and Rule 2.3 guarantee fundamental mode operation.

*Proposition 2.9:* Under Rule 2.3, the composite machine $\Sigma(C_A,C_D)$ of Figure 1 operates in fundamental mode. □

Appropriate controllers $C_A$ and $C_D$ can be constructed by a process similar to the one used by [5], [6], and [7].

*Construction 2.10:* **Building the controllers $C_A$ and $C_D$**: We describe the construction of $C_A$; the construction of $C_D$ is similar. The controller $C_A$ consists of two asynchronous trigger machines: an observer $\mathscr{O}$ and an action part $C_A^a$; here, $\mathscr{O}$ detects the end of a controller turn, while $C_A^a$ applies $A-$action commands to $\Sigma$, after being activated by $\mathscr{O}$.

*Part I: Constructing $\mathscr{O}$:* Choose two new and distinct characters $\chi_A$ and $\chi_D$ to serve as output characters of $\mathscr{O}$, where $\chi_A$ is to activate $C_A^a$ to start a $C_A$ turn, and $\chi_D$ is to activate $C_D^a$ to start a $C_D$ turn. In a $C_A$ turn starting at a state $x$ of $\Sigma$, $\mathscr{O}$ detects when $\Sigma$ stably reaches a state of $S_A(x)$ (see Lemma 2.8). In a $C_D$ turn starting at a state $x$ of $\Sigma$, $\mathscr{O}$ detects when $\Sigma$ stably reaches a state of $S_D(x)$. The output of $\mathscr{O}$ is generated by a function $\phi : X \times X \times \{A,D,N\} \to X \times \{\chi_A,\chi_D\} : (x,z,\zeta) \mapsto (x',\chi)$, where $x$ is the state of $\Sigma$ at the start of a controller turn; $z$ is the current state of $\Sigma$; $\zeta$ specifies the active controller ($A$ for $C_A$; $D$ for $C_D$; $N$ for no active controller); $x'$ is the state of $\Sigma$ at the end of the controller turn; and $\chi$ is $\chi_A$ or $\chi_D$, activating the next controller (below, $\backslash$ indicates set difference).

Initial turn ($x_0$ is the initial state of $\Sigma$):

$$\phi(x_0,x_0,N) := \begin{cases} \chi_A & \text{if } x_0 \notin T_A \cup T_D; \\ \neg & \text{otherwise.} \end{cases}$$

During a turn of $C_A$ that started at $x$:

$$\phi(x,z,A) := \begin{cases} (z,\chi_D) & \text{if } z \in S_A(x) \setminus T_A; \\ \neg & \text{otherwise.} \end{cases}$$

During a turn of $C_D$ that started $x$:

$$\phi(x,z,D) := \begin{cases} (z,\chi_A) & \text{if } z \in S_D(x) \setminus T_D; \\ \neg & \text{otherwise.} \end{cases}$$

The observer $\mathscr{O}$ remains silent after $\Sigma$ has reached a stable state at a target state.

*Part II: Building $C_A^a$:* Consider a turn of $C_A$ that starts at the state $x$ of $\Sigma$. Select a state $x' \in S_A(x)$ that satisfies the condition: there is an $A-$action string $u = u_1u_2\cdots u_{q_A(x)} \in (A\times\neg)^+$ that takes $\Sigma$ from $x$ to $x'$ without passing through a member of $S_A(x)$ (how to find such strings is discussed in later sections). This $A-$action will take $\Sigma$ through a string of states $x_1x_2\cdots x_{d_A(x)}$, where $x_{d_A(x)} = x'$; let $x_{i_1}, x_{i_2}, ..., x_{i_{q_A(x)}}$ be the stable states in this string, namely, $x_{i_1} = s(x,u_1)$, and $x_{i_{k+1}} = s(x_{i_k},u_{k+1})$, $k = 1,2,...,q_A(x)-1$, where $s$ is the stable recursion function of $\Sigma$ and $x_{i_{q_A(x)}} = x'$.

To implement the input string $u$, build in $C_A^a$ a subset of states $\Xi_A := \{\xi^0,\ \xi_A^1(x),\ \xi_A^2(x),\ ...,\ \xi_A^{q_A(x)}(x)\}$, where $\xi^0$ is the initial state of $C_A^a$. On these states, define the recursion function $\varphi_A : \Xi_A \times X \times \{\chi_A,\chi_D\} \to \Xi_A : (\xi,z,\chi) \mapsto$

$\varphi_A(\xi, z, \chi)$ by

$$\varphi_A(\xi, z, \chi) := \begin{cases} \xi^0 & \text{if } \xi = \xi^0, \chi \neq \chi_A; \\ \xi_A^1(x) & \text{if } \xi = \xi^0, z = x, \chi = \chi_A; \\ \xi_A^{k+1}(x) & \text{if } \xi = \xi_A^k(x), z = x_{i_k}, \chi = \neg, \\ & \quad k = 1, 2, ..., q_A(x) - 1; \\ \xi^0 & \text{if } \xi = \xi_A^{q_A(x)}(x), z = x', \chi = \neg; \end{cases}$$

and the output function $\eta_A : \Xi_A \to (A \times \neg)$ by

$$\eta_A(\xi) := \begin{cases} (\neg, \neg) & \text{if } \xi = \xi^0; \\ (u_k, \neg) & \text{if } \xi = \xi_A^{i_k}(x), k = 1, 2, ..., q_A(x). \end{cases}$$

It can then be seen that the resulting controller complies with Rule 2.3 (compare to [5] and [6]; see [23] for more details). □

## III. Transition Matrices

### A. Input pairs

The controlled machine $\Sigma$ has two inputs – one from the alphabet $A$ and one from the alphabet $D$ – and, as a result, its inputs are pairs $(u, \upsilon) \in A \times D$. In fundamental mode operation, only one input can be active at a time, i.e., $u = \neg$ or $\upsilon = \neg$, and inputs always come from the set

$$A \otimes D := \{(u, \upsilon) \in A \times D : u = \neg \text{ or } \upsilon = \neg)\}. \tag{3}$$

We use the character $N$ to denote an impossible transition.

*Concatenation* of strings $(a, b), (a', b') \in (A \otimes D)^+ \cup N$ is defined by

$$\text{conc}((a, b), (a', b')) := \begin{cases} (aa', bb') \text{ if } (a, b), (a', b') \in (A \otimes D)^+; \\ N \text{ if } (a, b) = N \text{ or } (a', b') = N. \end{cases}$$

For subsets $S_1, S_2 \subseteq (A \otimes D)^+ \cup N$, the *sum* is

$$S_1 \uplus S_2 := \begin{cases} S_1 \cup S_2 & \text{if } S_1 \neq N \text{ and } S_2 \neq N, \\ S_1 & \text{if } S_2 = N, \\ S_2 & \text{if } S_1 = N. \end{cases}$$

The *concatenation* of sets of strings $S_1, S_2 \subseteq (A \otimes D)^+ \cup N$ is

$$\text{conc}(S_1, S_2) \quad := \biguplus_{\sigma_1 \in S_1, \sigma_2 \in S_2} \text{conc}(\sigma_1, \sigma_2). \tag{4}$$

### B. Matrices

Generalizing a notion of [5] and [6], we build two $n \times n$ *one-step matrices of stable transitions*: for $i, j = 1, 2, ..., n$,

$$R_{ij}^1(\Sigma, A) := \begin{cases} \{(u, \neg) \in (A \times \neg) : x^j = s(x^i, (u, \neg))\} \\ \qquad\qquad \text{if } x^j \in s(x^i, (A \times \neg)), \\ N \qquad \text{otherwise}; \end{cases}$$

$$R_{ij}^1(\Sigma, D) := \begin{cases} \{(\neg, \upsilon) \in (\neg \times D) : x^j = s(x^i, (\neg, \upsilon))\} \\ \qquad\qquad \text{if } x^j \in s(x^i, (\neg \times D)), \\ N \qquad \text{otherwise}; \end{cases}$$

here, the first matrix describes outcomes of one step $A-$action and the second describes outcomes of one step $D-$action.

*Example 3.1:* For Example 2.4, a calculation yields

$$R^1(\Sigma, A) = \begin{pmatrix} \left\{\begin{matrix}(\neg,\neg), \\ (a^1,\neg)\end{matrix}\right\} & N & N & \{(a^2,\neg)\} & N \\ \{(a^1,\neg)\} & \{(\neg,\neg)\} & \{(a^2,\neg)\} & N & N \\ \left\{\begin{matrix}(a^1,\neg), \\ (a^2,\neg)\end{matrix}\right\} & N & \{(\neg,\neg)\} & N & N \\ \{(a^2,\neg)\} & N & N & \left\{\begin{matrix}(\neg,\neg), \\ (a^1,\neg)\end{matrix}\right\} & N \\ \{(a^1,\neg)\} & N & N & \{(a^2,\neg)\} & \{(\neg,\neg)\} \end{pmatrix} \square$$

For two such $n \times n$ matrices $E$ and $F$, the *sum* $E + F$ has the entries

$$(E + F)_{ij} := E_{ij} \uplus F_{ij}, i, j = 1, 2, ..., n;$$

and the *product* $EF$ has the entries

$$(EF)_{ij} \quad := \biguplus_{\ell = 1, ..., n} \text{conc}(E_{i\ell}, F_{\ell j}), i, j = 1, 2, ..., n.$$

Powers are defined by

$$E^\ell := EE^{\ell - 1}, \ell = 1, 2, 3, ...,$$

and the combined power is

$$E^{(\ell)} := E^1 + E^2 + \cdots + E^\ell.$$

We can now rephrase a result of [5] and [6].

*Proposition 3.2:* The following are true for two stable states $x^i, x^j$ of $\Sigma$:
*(i)* $x^j$ is stably reachable from $x^i$ through $A-$action if and only if $(R^1(\Sigma, A))_{ij}^{(n-1)} \neq N$.
*(ii)* $x^j$ is stably reachable from $x^i$ through $D-$action if and only if $(R^1(\Sigma, D))_{ij}^{(n-1)} \neq N$. □

We incorporate now the requirements of Rule 2.3 into the transitions matrices: (a string $u^1$ is a *strict prefix* of a string $u^2$ if there is a non-empty string $u^3$ such that $u^2 = u^1 u^3$).

*Construction 3.3:* **The matrix of $A-$stable transitions:** On the matrix $(R^1(\Sigma, A))^{(n-1)}$, perform the following operations for $i = 1, 2, ..., n$ (here, $T_A, T_D$ are the target sets and $\omega$ is the weight function):

**Step 1:** If $x^i \in T_A \cup T_D$, then replace all off-diagonal entries of row $i$ by $N$.

**Step 2:** If $x^i \in T_D$, then replace by $N$ all off-diagonal entries of column $i$.

**Step 3:** Denote by $\zeta_A(i)$ the set of all remaining integers $j \in \{1, 2, ..., n\}$ for which position $j$ of row $i$ is not $N$. If $\zeta_A(i) \neq \varnothing$, the minimal weight of a stably reachable state is $w_A(i) = \min_{j \in \zeta_A(i)} \omega(x^j)$. Replace by $N$ all entries $j$ of row $i$ for which $\omega(x^j) > w_A(i)$.

**Step 4:** In row $i$, delete all strings that include as a strict prefix a string that appears anywhere else in row $i$; replace by $N$ entries that become empty.

**Step 5:** If an entry includes the string $(\neg, \neg)$, then delete all other strings from this entry.

This yields the *matrix of $A-$stable transitions* $R(\Sigma, A)$. □

*Example 3.4:* For Example 2.4, a direct calculation yields: (only one string is listed per entry to simplify typography)

$$R(\Sigma, A) = \begin{pmatrix} N & N & N & \{(a^2,\neg)\} & N \\ N & N & \{(a^2,\neg)\} & N & N \\ N & N & \{(\neg,\neg)\} & N & N \\ N & N & N & \{(\neg,\neg)\} & N \\ N & N & N & N & \{(\neg,\neg)\} \end{pmatrix}. \quad \square$$

The $A-$*action skeleton matrix* $K(\Sigma,A)$ is then:

$$K_{ij}(\Sigma,A) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma,A) \neq N, \\ 0 & \text{otherwise.} \end{cases} \quad j = 1,2,...,n, \quad (5)$$

*Example 3.5:* From Example 3.4:

$$K(\Sigma,A) = \begin{pmatrix} 0&0&0&1&0 \\ 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix}. \qquad \square$$

The following is a consequence of Construction 3.3 and (5).

*Proposition 3.6:* Under Rule 2.3, one turn of $A-$action can take $\Sigma$ from a state $x^i$ to a state $x^j$ if and only if $K_{ij}(\Sigma,A) = 1$. In that case, $R_{ij}(\Sigma,A)$ consists of $A-$action input strings that take $\Sigma$ from $x^i$ to $x^j$, observing Rule 2.3.□

The *matrix of* $D-$*stable transitions* $R(\Sigma,D)$ and the $D-$*action skeleton matrix* $K(\Sigma,D)$ are constructed analogously and have features similar to those of $R(\Sigma,A)$ and $K(\Sigma,A)$, respectively.

A *terminal state* is a state from which a machine cannot be moved. The definition of a skeleton matrix implies:

*Proposition 3.7:* Under Rule 2.3, the following are valid for a state $x^j$ of $\Sigma$. *(i)* $x^j$ is a terminal state for $A-$action if and only if $K_{jj}(\Sigma,A) = 1$. *(ii)* $x^j$ is a terminal state for $D-$action if and only if $K_{jj}(\Sigma,D) = 1$. □

## IV. Consecutive Turns

By Rule 2.3, the controllers $C_A$ and $C_D$ of Figure 1 operate in turns: $C_A$ starts from the initial state $x_0 = x^i$ of $\Sigma$; by Proposition 3.6, it takes $\Sigma$ to a state $x^j$ selected by the designer from among the states for which $R_{ij}(\Sigma,A) \neq N$. The input string $C_A$ must generate for $\Sigma$ is a member of that $R_{ij}(\Sigma,A)$ entry, and $C_A$ is built by Construction 2.10.

Next comes a turn of $C_D$. The states $\Sigma$ can stably reach at the end of this turn are given by the non-$N$ entries of row $i$ of $R(\Sigma,A)R(\Sigma,D)$; designer selection determines which one these will be implemented. Next, $C_A$ acts again; at the end of its turn, $\Sigma$ can rest at any state for which the corresponding entry of row $i$ of $R(\Sigma,A)R(\Sigma,D)R(\Sigma,A)$ is not $N$. Continuing in this way leads to the following.

*Definition 4.1:* The *compound matrix of stable transitions* $R(\Sigma)$ and the *compound skeleton matrix* $K(\Sigma)$:

$$R(\Sigma) := \sum_{i=1}^{n-1} \left[ (R(\Sigma,A)R(\Sigma,D))^{i-1} R(\Sigma,A) + (R(\Sigma,A)R(\Sigma,D))^{i} \right]$$

$$K_{ij}(\Sigma) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma) \neq N, \\ 0 & \text{otherwise.} \end{cases} \qquad \square$$

*Example 4.2:* For the machine $\Sigma$ of Example 2.4,

$$\begin{aligned} R(\Sigma) \;=\; & R(\Sigma,A) + (R(\Sigma,A)R(\Sigma,D)) + \\ & (R(\Sigma,A)R(\Sigma,D))R(\Sigma,A) + (R(\Sigma,A)R(\Sigma,D))^2 + \\ & (R(\Sigma,A)R(\Sigma,D))^2 R(\Sigma,A) + (R(\Sigma,A)R(\Sigma,D))^3 + \\ & (R(\Sigma,A)R(\Sigma,D))^3 R(\Sigma,A) + (R(\Sigma,A)R(\Sigma,D))^4 . \square \end{aligned}$$

The fact that $\Sigma$ has only $n$ states enables us to prove the following (see [23] for details).

*Lemma 4.3:* Assume that $\Sigma$ starts from the initial state $x_0 = x^i$ and is controlled in compliance with Rule 2.3. A state $x^j$ of $\Sigma$ can be the outcome of a succession of controller turns if and only if $K_{ij}(\Sigma) = 1$. □

This allows us to characterize the final outcomes of the control process of Figure 1.

*Theorem 4.4:* Assume that the control configuration of Figure 1 operates in accord with Rule 2.3, with $\Sigma$ having the initial state $x_0 = x^i$. Then,

*(i)* There are controllers $C_A$ and $C_D$ that guide $\Sigma$ to a terminal state at $x^j$ if and only if $K_{ij}(\Sigma) = 1$, $K_{jj}(\Sigma,A) = 1$ and $K_{jj}(\Sigma,D) = 1$.

*(ii)* There are controllers $C_A$ and $C_D$ that guide $\Sigma$ into an infinite cycle if and only if there are integers $p \neq q \in \{1,2,...,n\}$ such that $K_{ip}(\Sigma) = 1$, $K_{pq}(\Sigma) = 1$ and $K_{qp}(\Sigma) = 1$.

*Proof: (sketch)* By Lemma 4.3, the state $x^j$ can be reached if and only if $K_{ij}(\Sigma) = 1$. By Proposition 3.7, the remaining conditions of *(i)* are required for $x^j$ to be a terminal state. Regarding *(ii)*, Lemma 4.3 also shows that controllers that force transitions from $x^p$ to $x^q$ and back from $x^q$ to $x^p$ exist if and only if $K_{pq}(\Sigma) = 1$, $K_{qp}(\Sigma) = 1$. ■

Theorem 4.4 yields the following characterization of the set $\Theta$ of all possible terminal states of $\Sigma$ in Figure 1.

*Corollary 4.5:* Under the conditions of Theorem 4.4:

*(i)* $\Theta = \left\{ x^j \in X : K_{ij}(\Sigma) = 1, K_{jj}(\Sigma,A) = 1, K_{jj}(\Sigma,D) = 1 \right\}$.

*(ii)* All controllers $C_A$ and $C_D$ take $\Sigma$ to a terminal state if and only if $x^q \in \Theta$ whenever $K_{iq}(\Sigma) = 1$.

*(iii)* $\Sigma$ may enter an infinite cycle for some controllers $C_A$ and $C_D$ if and only if $K_{iq}(\Sigma) = 1$ for some $x^q \notin \Theta$. □

This allows us to determine when a controller can prevail:

*Corollary 4.6:* *(i)* $C_A$ can prevails for some designs of $C_D$ if and only if $K_{ij}(\Sigma) = 1$ for some $j \in T_A$.

*(ii)* Every $C_A$ prevails for any design of $C_D$ if and only if $j \in T_A$ whenever $K_{ij}(\Sigma) = 1$.

*(iii)* $C_D$ can prevail for some designs of $C_A$ if and only if $K_{ij}(\Sigma) = 1$ for some $j \in T_D$.

*(iv)* Every $C_D$ prevails for any design of $C_A$ if and only if $j \in T_D$ whenever $K_{ij}(\Sigma) = 1$. □

Thus, skeleton matrices allow us to determine all outcomes of the control process. Yet, the design of controllers does require matrices of stable transitions, as these provide the strings for controller implementation in Construction 2.10.

## V. States of Certainty

For some states of $\Sigma$, the outcome of the control process becomes pre-determined: one controller prevails if properly designed, irrespective of actions taken by the other controller.

*Definition 5.1:* In Figure 1, the controller $C_A$ (respectively, $C_D$) can *prevail with certainty* from a state $x$ of $\Sigma$ if $C_A$ (respectively, $C_D$) can always prevail after starting a turn at $x$, irrespective of actions taken by the other controller. □

We need some notation. Let $\chi : X \to \{0,1\}^n$ be the function that assigns to a set $S = \{x^{i_1}, x^{i_2}, ..., x^{i_q}\}$ of states of $\Sigma$ the column vector $\chi(S) = (0,...,1,0,...,0,1,0,...,0)^T$, where 1 appears in positions $i_1, i_2, ..., i_q$ and zeros everywhere else; the empty set of states is represented by the zero vector. An

$n \times n$ skeleton matrix $K$ operates on a vector $t \in \{0,1\}^n$ with the result $t' = Kt$, where $t' = (t'_1, t'_2, ..., t'_n)^T$ and

$$t'_i := \max_{q=1,2,...,n} \{K_{iq} t_q\},\ i = 1,2,...,n.$$

To add vectors $\vartheta = (\vartheta_1, ..., \vartheta_n)^T, \vartheta' = (\vartheta'_1, ..., \vartheta'_n)^T \in \{0,1\}^n$:

$$(\vartheta + \vartheta')_i := \max\{\vartheta_i, \vartheta'_i\},\ i = 1,2,...,n. \qquad (6)$$

By definition of $K(\Sigma, A)$, the set $S(j)$ of all states of $\Sigma$ from which the state $x^j$ is stably reachable in one turn of $A-$action is given by all entries of 1 in column $j$ of $K(\Sigma, A)$. Consequently, $\chi(S(j)) = K(\Sigma, A)\chi(x^j)$. More generally,

*Proposition 5.2:* Let $S$ be a set of states of $\Sigma$. The set of all states of $\Sigma$ from which a member of $S$ is stably reachable in one turn of $A-$action (respectively, $D-$action) is given by the vector $K(\Sigma, A)\chi(S)$ (respectively, $K(\Sigma, D)\chi(S)$). □

By Proposition 5.2, the set of all states of $\Sigma$ from which $C_A$ can prevail in one turn is given by $v_A^1 := K(\Sigma, A)\chi(T_A)$.

The *complement* $v^c$ of a vector $v \in \{0,1\}^n$ is obtained by replacing every 0 by 1 and every 1 by 0. Then, $(v_A^1)^c$ indicates the set of all states of $\Sigma$ from which a member of $T_A$ is not stably reachable in one turn of $A-$action. Thus, $\theta_A^1 := K(\Sigma, D)(v_A^1)^c$ characterizes all states of $\Sigma$ from which one turn of $D-$action can prevent $\Sigma$ from entering the set $v_A^1$; so $\theta_A^1$ is the set of all states of $\Sigma$ from which $C_D$ can prevent $C_A$ from prevailing in one turn. Hence, $v_A^2 := (\theta_A^1)^c = (K(\Sigma, D)(v_A^1)^c)^c$ indicates all states of $\Sigma$ from which $D-$action cannot block $C_A$ from prevailing in one turn. Thus, if $\Sigma$ is in a state represented in $v_A^2$, then $C_A$ can prevail, irrespective of actions taken by $C_D$. Continuing in this manner we obtain the set of all states of $\Sigma$ from which $C_A$ can prevail with certainty (see [23] for details):

$$\begin{aligned}
v_A^0 &:= \chi(T_A) \\
v_A^1 &:= K(\Sigma, A)\chi(T_A) \\
v_A^i &:= \left[K(\Sigma, D)(v_A^{i-1})^c\right]^c \text{ for } i = 2,4,6,... \\
v_A^i &:= K(\Sigma, A)v_A^{i-1} \text{ for } i = 3,5,7,...
\end{aligned}$$

$$v_A := \sum_{i=0}^{n-1} v_A^i. \qquad (7)$$

*Theorem 5.3:* The set of all states of $\Sigma$ from which $C_A$ can prevail with certainty is given by $v_A$. □

Needless to say, for $C_A$ to prevail from a state of certainty it must be properly designed; appropriate strings that $C_A$ must generate are taken from entries of $R(\Sigma, A)$ and used in Construction 2.10 to build $C_A$.

*Example 5.4:* For Example 2.4, $v_A = (0,1,1,0,0)^T$, so that $C_A$ prevails with certainty from $x^2$ and $x^3$. □

A slight reflection leads to the following simple condition.

*Corollary 5.5:* $C_D$ can block $C_A$ from prevailing if and only if the initial condition $\mathrm{x}_0$ of $\Sigma$ corresponds to an entry of 0 in $v_A$. □

Analogous results can be obtained for the other controller.

## REFERENCES


[1] J. Sparso and S. Furber, *Principles of Asynchronous Circuit Design – A Systems Perspective*. Dordrecht, the Netherlands: Kluwer Academic Publishers, 2001.

[2] J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089–1120, 2006.

[3] R. F. Tinder, *Asynchronous Sequential Machine Design and Analysis*. London, UK: Morgan and Claypool Publishers, 2009.

[4] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill Book Company, 1970.

[5] T. E. Murphy, X. Geng, and J. Hammer, "Controlling races in asynchronous sequential machines," in *Proceeding of the IFAC World Congress*, July 2002.

[6] ——, "On the control of asynchronous machines with races," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1073–1081, 2003.

[7] X. Geng and J. Hammer, "Input/output control of asynchronous sequential machines," *IEEE Transactions on Automatic Control*, vol. 50, no. 12, pp. 1956–1970, 2005.

[8] N. Venkatraman and J. Hammer, "Stable realizations of asynchronous sequential machines with infinite cycles," in *Proceeding of the Asian Control Conference*, Bali, Indonesia, 2006, pp. 45–51.

[9] ——, "On the control of asynchronous sequential machines with infinite cycles," *International Journal of Control*, vol. 79, no. 7, pp. 764–785, 2006.

[10] J. M. Yang and J. Hammer, "State feedback control of asynchronous sequential machines with adversarial inputs," *International Journal of Control*, vol. 81, no. 12, pp. 1910–1929, 2008.

[11] ——, "Asynchronous sequential machines with adversarial intervention: the use of bursts," *International Journal of Control*, vol. 83, no. 5, pp. 956–969, 2010.

[12] J. Peng and J. Hammer, "Input/output control of asynchronous sequential machines with races," *International Journal of Control*, vol. 83, no. 1, pp. 125–144, 2010.

[13] ——, "Bursts and output feedback control of non-deterministic asynchronous sequential machines," *European Journal of Control*, vol. 18, no. 3, pp. 286–300, 2012.

[14] J. M. Yang, "Model matching inclusion for input/state asynchronous sequential machines," *Automatica*, vol. 47, no. 3, pp. 597–602, 2011.

[15] J. M. Yang and S. W. Kwak, "Realizing fault-tolerant asynchronous sequential machines using corrective control," *IEEE Transactions on Control System Technology*, vol. 18, no. 6, pp. 1457–1463, 2010.

[16] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," *SIAM Journal on Control and Optimization*, vol. 32, no. 4, pp. 1075–1097, 1994.

[17] J. Hammer, "On some control problems in molecular biology," in *Proceedings of the IEEE conference on Decision and Control*, December 1994, pp. 4098–4103.

[18] ——, "On corrective control of sequential machines," *International Journal of Control*, vol. 65, no. 2, pp. 249–276, 1996.

[19] R. Kumar, S. Nelvagal, and S. I. Marcus, "A discrete event systems approach for protocol conversion," *Discrete Event Systems: Theory and Applications*, vol. 7, no. 3, pp. 295–315, 1997.

[20] G. Barrett and S. Lafortune, "Bisimulation, the supervisory control problem, and strong model matching for finite state machines," *Discrete Event Systems: Theory and Applications*, vol. 8, no. 4, pp. 377–429, 1998.

[21] M. D. Di-Benedetto, A. Sangiovanni-Vincentelli, and T. Villa, "Model matching for finite-state machines," *IEEE Transactions on Automatic Control*, vol. 46, no. 11, pp. 1726–1743, 2001.

[22] N. Yevtushenko, T. Villa, R. K. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli, "Compositionally progressive solutions of synchronous fsm equations," *Discrete Event Systems: Theory and Applications*, vol. 18, no. 4, pp. 51–89, 2008.

[23] J. Hammer, "Automatic defensive control of asynchronous sequential machines," *Submitted for publication*, 2014.